



## Tarea Computacional I

### 1. Implementación de la Factorización $LU$

Implementar cada una de los siguientes algoritmos del Álgebra Lineal en MATLAB empleando solo comandos básicos (no emplear las funciones intrínsecas que posee MATLAB). Debe usar las cabeceras de función dada para cada una.

- a) Factorización  $LU$  sin pivoteo, sin embargo el algoritmo debe verificar por pivote cero y finalizar emitiendo el respectivo mensaje de error. El código debe retornar las matrices  $L$  y  $U$  como una sola matriz con la idea de ahorrar espacio en memoria.

```
function LU = LUFactorization(A)
% Uso: LU = LUFactorization(A)
% Computa la factorización LU de la matriz A
% y retorna la matriz factorizada
```

- b) Sustitución Progresiva. Esta rutina debe tomar como dato de entrada la matriz resultante de la Factorización  $LU$  de la parte (a)

```
function y = ForwardSubstitution(LU, b)
% Uso: y = ForwardSubstitution(LU, b)
% Lleva a cabo sustitución progresiva usando la parte
% triangular inferior unitaria de la matriz LU.
```

- c) Sustitución Regresiva. Esta rutina debe tomar como dato de entrada la matriz resultante de la Factorización  $LU$  de la parte (a)

```
function x = BackwardSubstitution(LU, y) % Uso: y = ForwardSubstitution(LU, b)
% Lleva a cabo sustitución regresiva usando la parte
% triangular superior de la matriz LU.
```

- d) Use el archivo `Verify.m` para probar sus códigos. Seleccione un entero  $n$  entre 100 y 200, ejecute `VerifyLU(n)` e indique para  $n$  el error obtenido.

### 2. Tridiagonal $LU$

Considere la siguiente matriz tridiagonal

$$A_t = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & & 0 \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & & & a_n & b_n \end{pmatrix}_{n \times n}$$

Tales sistemas pueden encontrarse en distintas aplicaciones como por ejemplo Interpolación por Splines. Asumiendo que no hay necesidad de pivoteo, entonces los factores de  $A_t = LU$  tienen la forma

$$L = \begin{pmatrix} 1 & 0 & & \cdots & 0 & 0 \\ l_2 & 1 & 0 & & 0 & 0 \\ 0 & l_3 & 1 & & & 0 \\ & & \ddots & \ddots & & \\ 0 & 0 & & & 1 & 0 \\ 0 & 0 & & & l_n & 1 \end{pmatrix} \quad U = \begin{pmatrix} d_1 & u_1 & & \cdots & 0 & 0 \\ 0 & d_2 & u_2 & & & 0 \\ 0 & 0 & d_3 & u_3 & & \\ & & & \ddots & \ddots & \\ 0 & 0 & & & d_{n-1} & u_{n-1} \\ 0 & 0 & & & 0 & d_n \end{pmatrix}$$

- Derive la relación de recurrencia para  $l_i$ ,  $d_i$  y  $u_i$  en términos de  $a_i$ ,  $b_i$  y  $c_i$ . Explique su trabajo.
- Implemente la rutina que permita resolver  $A_t x = b$  usando la descomposición  $LU$  derivada en su trabajo. La rutina debe tomar como argumentos 4 arreglos unidimensionales, tres correspondientes a las diagonales de  $A_t$  y el otro correspondiente al vector de términos independientes. `function x = TriDiagonalSolve(a, b, c, k)`  
`% Uso: x = TriDiagonalSolve(a, b, c, k)`  
`% Resuelve el sistema Ax = k, donde`  
`% A es una matriz tridiagonal. Los vectores columna`  
`% a, b, c describen las entradas de las diagonales de`  
`% la matriz. a(1) y c(n) son ignorados.`
- ¿Cuál es la complejidad computacional de su algoritmo?, Explique.
- Use el archivo `VerifyTridiagonalLU.m` para probar sus códigos. Seleccione un entero  $n$  entre 1000 y 2000, ejecute `VerifyTridiagonalLU(n)` e indique para  $n$  el error obtenido.

### 3. Sistemas Lineales para Ecuaciones en Diferencia.

Sistemas de ecuaciones lineales frecuentemente se generan ecuaciones diferenciales en computación. Considere el siguiente problema de ecuaciones diferenciales:

$$u'' = f(t)u(0) = 0u(1) = 0 \quad (1)$$

con

$$f(t) = 16\pi \cos(8\pi t^2) - 256\pi^2 t^2 \sin(8\pi t^2), \quad t \in [0, 1]$$

La solución exacta a este problema está dada por  $u(t) = \sin(8\pi t^2)$ . Aproximando la solución exacta  $u$  a este sistema en un número discreto de puntos, se puede construir una aproximación  $\{v_i\}_{i=0}^N$  tal que  $v_0 = u_0 = 0$  cuando  $t_0 = 0$  y  $v_N = u_N = 0$  cuando  $t_N = 1$ . Aplicando una aproximación en Diferencia Finitas dada por:

$$u''_i \approx \frac{v_{i-1} - 2v_i + v_{i+1}}{h^2}$$

donde  $h$  representa el tamaño de paso ( $h = 1/N$ ). Se define el tiempo en cada punto como  $t_i = i/N$ . Con esta aproximación se genera un sistema tridiagonal  $(N-1) \times (N-1)$  de la forma:

$$\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -2 & 1 \\ 0 & 0 & 0 & \cdots & 1 & -2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{N-2} \\ v_{N-1} \end{pmatrix} = h^2 \begin{pmatrix} f(t_1) \\ f(t_2) \\ f(t_3) \\ \vdots \\ f(t_{N-2}) \\ f(t_{N-1}) \end{pmatrix} - \begin{pmatrix} v_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ v_N \end{pmatrix}$$

donde  $v_0$  y  $v_N$  son definidos en el problema. Sea  $n = N - 1$  el número de incógnitas en el sistema lineal

- a) Resolver este sistema usando la descomposición  $LU$  y Sustitución progresiva y regresiva con  $n = 10, 100, 200, 400, 800$ . Determine el cantidad de tiempo computacional requerido para cada  $n$  (Use `tic` y `toc` para este propósito y `format long` para ver la cantidad de dígitos suficientes). Compare los resultados en cada paso con la solución exacta en la misma gráfica tanto para  $n = 10$  y  $n = 400$ . Explique que observa.
- b) Resolver este sistema usando la descomposición  $LU$  para sistemas tridiagonales con  $n = 10, 100, 200, 400, 800$ . Determine el cantidad de tiempo computacional requerido para cada  $n$ . Compare los resultados en cada paso con la solución exacta en la misma gráfica tanto para  $n = 10$  y  $n = 400$ .
- c) Compare los resultados obtenidos en los dos items anteriores. ¿Cuál algoritmo provee una solución del sistema más rápido?. Muestre una tabla comparativa de los tiempos de ejecución para cada valor de  $n$ . Los resultados coinciden con la complejidad computacional de cada algoritmo?